**INTERACTIVE PROJECTIONS**

*A Project Report*

*submitted by*

**Bhogi SriHarsha 14BCE1127**

*Under guidance of* ***Prof. Rajesh Kanna***

*in partial fulfillment of the requirements*

*for the award of the degree of*

**BACHELOR OF TECHNOLOGY***in***COMPUTER SCIENCE AND ENGINEERING**

****

**SCHOOL OF COMPUTER SCIENCE & ENGINEERING**

**VIT UNIVERSITY**

# AIM

To show the orthographic and perspective projections interactively in such a way that the user will have the full control of the navigation in the 3d world and to represent the 3d world using Object Oriented Programming.

# ANALYSIS

**REPRESENTATION OF 3D WORLD IN OOPS**

Class for point: class point {

float x,y,z;

};

Class for line: class lyne {

float start,end;

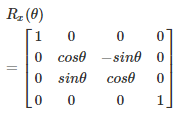
};

// the word ‘line’ is not possible as it is built-in

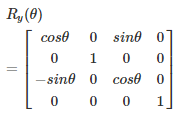
**ORTHOGRAPHIC PROJECTIONS**

A dynamic projection matrix is used for the orthographic projection at various angles around x-axis and y-axis.

Projection matrix for rotation around X-axis:



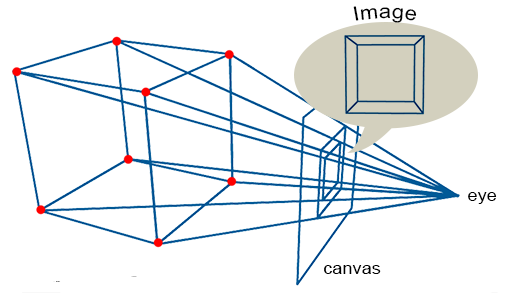
Projection matrix for rotation around Y-axis:



The projection matrices are multiplied to the points and concatenated to each other using user defined matrix multiplication functions.

The program will run in a loop in which users can rotate the view in the direction they wish using w, a, s & d keys.

**PERSPECTIVE PROJECTIONS**



*Perspective projection of a cube on a canvas*

# CODE

#include<iostream>

#include<graphics.h>

#include<conio.h>

#include<math.h>

using namespace std;

class point

{

public:

float x,y,z;

};

class lyne

{

public:

point start,end;

};

class matrix4

{

public:

float a11,a12,a13,a14;

float a21,a22,a23,a24;

float a31,a32,a33,a34;

float a41,a42,a43,a44;

};

void printmatrix(matrix4 a)

{

cout<<a.a11<<" "<<a.a12<<" "<<a.a13<<" "<<a.a14<<endl;

cout<<a.a21<<" "<<a.a22<<" "<<a.a23<<" "<<a.a24<<endl;

cout<<a.a31<<" "<<a.a32<<" "<<a.a33<<" "<<a.a34<<endl;

cout<<a.a41<<" "<<a.a42<<" "<<a.a43<<" "<<a.a44<<endl;

}

matrix4 multiply(matrix4 a,matrix4 b)

{

matrix4 idi;

idi.a11=a.a11\*b.a11+a.a12\*b.a21+a.a13\*b.a31+a.a14\*b.a41;

idi.a12=a.a11\*b.a12+a.a12\*b.a22+a.a13\*b.a32+a.a14\*b.a42;

idi.a13=a.a11\*b.a13+a.a12\*b.a23+a.a13\*b.a33+a.a14\*b.a43;

idi.a14=a.a11\*b.a14+a.a12\*b.a24+a.a13\*b.a34+a.a14\*b.a44;

idi.a21=a.a21\*b.a11+a.a22\*b.a21+a.a23\*b.a31+a.a24\*b.a41;

idi.a22=a.a21\*b.a12+a.a22\*b.a22+a.a23\*b.a32+a.a24\*b.a42;

idi.a23=a.a21\*b.a13+a.a22\*b.a23+a.a23\*b.a33+a.a24\*b.a43;

idi.a24=a.a21\*b.a14+a.a22\*b.a24+a.a23\*b.a34+a.a24\*b.a44;

idi.a31=a.a31\*b.a11+a.a32\*b.a21+a.a33\*b.a31+a.a34\*b.a41;

idi.a32=a.a31\*b.a12+a.a32\*b.a22+a.a33\*b.a32+a.a34\*b.a42;

idi.a33=a.a31\*b.a13+a.a32\*b.a23+a.a33\*b.a33+a.a34\*b.a43;

idi.a34=a.a31\*b.a14+a.a32\*b.a24+a.a33\*b.a34+a.a34\*b.a44;

idi.a41=a.a41\*b.a11+a.a42\*b.a21+a.a43\*b.a31+a.a44\*b.a41;

idi.a42=a.a41\*b.a12+a.a42\*b.a22+a.a43\*b.a32+a.a44\*b.a42;

idi.a43=a.a41\*b.a13+a.a42\*b.a23+a.a43\*b.a33+a.a44\*b.a43;

idi.a44=a.a41\*b.a14+a.a42\*b.a24+a.a43\*b.a34+a.a44\*b.a44;

return idi;

}

point multiply(point b,matrix4 a)

{

point idi;

float xx=idi.x,yy=idi.y,zz=idi.z;

float x=b.x,y=b.y,z=b.z;

xx=x\*a.a11+y\*a.a21+z\*a.a31+a.a41;

yy=x\*a.a12+y\*a.a22+z\*a.a32+a.a42;

zz=x\*a.a13+y\*a.a23+z\*a.a33+a.a43;

idi.x=xx; idi.y=yy; idi.z=zz;

return idi;

}

matrix4 newmatrix(float a11,float a12,float a13,float a14, float a21,float a22,float a23,float a24, float a31,float a32,float a33,float a34, float a41,float a42,float a43,float a44)

{

matrix4 i;

i.a11=a11; i.a12=a12; i.a13=a13; i.a14=a14;

i.a21=a21; i.a22=a22; i.a23=a23; i.a24=a24;

i.a31=a31; i.a32=a32; i.a33=a33; i.a34=a34;

i.a41=a41; i.a42=a12; i.a43=a13; i.a44=a44;

return i;

}

float theta=0,omega=0;

//PROJECTION MATRICES

matrix4 ex=newmatrix(1,0,0,0, 0,cos(theta),sin(theta),0, 0,-sin(theta),cos(theta),0, 0,0,0,1); //rotation matrix around y axis

matrix4 wy=newmatrix(cos(omega),0,-sin(omega),0, 0,1,0,0, sin(omega),0,cos(omega),0, 0,0,0,1); //rotation matrix around y axis

matrix4 tr=newmatrix(1,0,0,0, 0,1,0,0, 0,0,1,0, 300,200,200,1); //translation matrix

point newpoint(float x1,float y1,float z1)

{

point idi;

idi.x=x1;

idi.y=y1;

idi.z=z1;

return idi;

}

lyne newlyne(float x1,float y1,float z1,float x2,float y2,float z2)

{

lyne idi;

idi.start.x=x1;

idi.start.y=y1;

idi.start.z=z1;

idi.end.x=x2;

idi.end.y=y2;

idi.end.z=z2;

return idi;

}

int orthographic() //ORTHOGRAPHIC PROJECTION

{

int i;

char opt;

float x1,y1,x2,y2;

point p1,p2;

lyne lyn[12];

lyn[0]=newlyne(-100,-100,-100,100,-100,-100);

lyn[1]=newlyne(-100,-100,-100,-100,-100,100);

lyn[2]=newlyne(-100,-100,-100,-100,100,-100);

lyn[3]=newlyne(100,100,100,-100,100,100);

lyn[4]=newlyne(100,100,100,100,-100,100);

lyn[5]=newlyne(100,100,100,100,100,-100);

lyn[6]=newlyne(100,100,-100,100,-100,-100);

lyn[7]=newlyne(-100,100,-100,100,100,-100);

lyn[8]=newlyne(-100,100,-100,-100,100,100);

lyn[9]=newlyne(-100,100,100,-100,-100,100);

lyn[10]=newlyne(-100,-100,100,100,-100,100);

lyn[11]=newlyne(100,-100,-100,100,-100,100);

printmatrix(ex);

do

{

for(i=0;i<12;i++)

{

p1.x=lyn[i].start.x;

p1.y=lyn[i].start.y;

p1.z=lyn[i].start.z;

p2.x=lyn[i].end.x;

p2.y=lyn[i].end.y;

p2.z=lyn[i].end.z;

p1=multiply(p1,multiply(ex,wy));

p2=multiply(p2,multiply(ex,wy));

//p1=multiply(p1,tr);

//p2=multiply(p2,tr);

line(p1.x+300,p1.y+200,p2.x+300,p2.y+200);

}

opt=getch();

if(opt==97) omega+=M\_PI/8;

else if(opt==100) omega-=M\_PI/8;

else if(opt==119) theta+=M\_PI/6;

else if(opt==115) theta-=M\_PI/6;

ex=newmatrix(1,0,0,0, 0,cos(theta),sin(theta),0, 0,-sin(theta),cos(theta),0, 0,0,0,1);

wy=newmatrix(cos(omega),0,-sin(omega),0, 0,1,0,0, sin(omega),0,cos(omega),0, 0,0,0,1);

cout<<sin(theta)<<endl;

printmatrix(ex);

cleardevice();

}

while(opt==97||opt==100||opt==119||opt==115); //a d w s ke ascii values

getch();

return 0;

}

int perspective() //PERSPECTIVE PROJECTION

{

int i;

point per=newpoint(0,0,-500); //initial perspective point

char opt;

int x31,y31,x32,y32;

float s31,w31,s32,w32;

float x1,y1,z1,x2,y2,z2;

float zed=0;

point p1,p2;

lyne lyn[12];

lyn[0]=newlyne(-100,-100,-100,100,-100,-100);

lyn[1]=newlyne(-100,-100,-100,-100,-100,100);

lyn[2]=newlyne(-100,-100,-100,-100,100,-100);

lyn[3]=newlyne(100,100,100,-100,100,100);

lyn[4]=newlyne(100,100,100,100,-100,100);

lyn[5]=newlyne(100,100,100,100,100,-100);

lyn[6]=newlyne(100,100,-100,100,-100,-100);

lyn[7]=newlyne(-100,100,-100,100,100,-100);

lyn[8]=newlyne(-100,100,-100,-100,100,100);

lyn[9]=newlyne(-100,100,100,-100,-100,100);

lyn[10]=newlyne(-100,-100,100,100,-100,100);

lyn[11]=newlyne(100,-100,-100,100,-100,100);

printmatrix(ex);

do

{

for(i=0;i<12;i++)

{

p1.x=lyn[i].start.x;

p1.y=lyn[i].start.y;

p1.z=lyn[i].start.z;

p2.x=lyn[i].end.x;

p2.y=lyn[i].end.y;

p2.z=lyn[i].end.z;

p1=multiply(p1,multiply(ex,wy));

p2=multiply(p2,multiply(ex,wy));

//p1=multiply(p1,tr);

//p2=multiply(p2,tr);

s31=(p1.z\*per.x-p1.x\*per.z+zed\*(p1.x-per.x))/(p1.z-per.z);

w31=(p1.z\*per.y-p1.y\*per.z+zed\*(p1.y-per.x))/(p1.z-per.z);

s32=(p2.z\*per.x-p2.x\*per.z+zed\*(p2.x-per.x))/(p2.z-per.z);

w32=(p2.z\*per.y-p2.y\*per.z+zed\*(p2.y-per.x))/(p2.z-per.z);

p1.x=s31;

p1.y=w31;

p2.x=s32;

p2.y=w32;

line(p1.x+300,p1.y+200,p2.x+300,p2.y+200);

}

opt=getch();

if(opt==97) omega+=M\_PI/8;

else if(opt==100) omega-=M\_PI/8;

else if(opt==119) theta+=M\_PI/6;

else if(opt==115) theta-=M\_PI/6;

else if(opt==43)

{

zed+=50;

per.z+=50;

}

else if(opt==45)

{

zed-=50;

per.z-=50;

}

ex=newmatrix(1,0,0,0, 0,cos(theta),sin(theta),0, 0,-sin(theta),cos(theta),0, 0,0,0,1);

wy=newmatrix(cos(omega),0,-sin(omega),0, 0,1,0,0, sin(omega),0,cos(omega),0, 0,0,0,1);

cout<<sin(theta)<<endl;

printmatrix(ex);

cleardevice();

}

while(opt==97||opt==100||opt==119||opt==115||opt==43||opt==45); //a d w s + - ke ascii values

getch();

return 0;

}

int main()

{

int gd = DETECT,gm;

char opt;

int y=0;

initgraph(&gd, &gm, "C:\\TC\\BGI");

cout<<"("<<getmaxx()<<","<<getmaxy()<<")";

//settextstyle(1,HORIZ\_DIR,0);

do {

outtextxy(220,120,"ORTHOGRAPHIC PROJECTION");

outtextxy(220,195,"PERSPECTIVE PROJECTION");

outtextxy(220,270,"EXIT THE APPLICATION");

rectangle(165,100+y,465,150+y);

opt=getch();

if(opt==115)

{

y+=75;

}

else if(opt==119)

{

y-=75;

}

if(y<0) y=150;

if(y>150) y=0;

cleardevice();

}

while(opt!=13);

if(y==0) orthographic();

if(y==75) perspective();

if(y==150) exit(0);

getch();

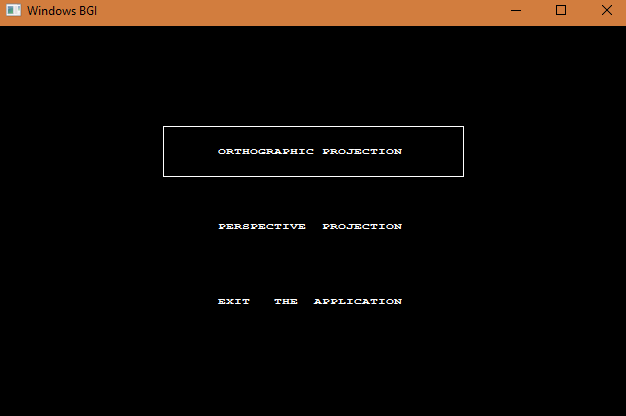
closegraph();

return 0;

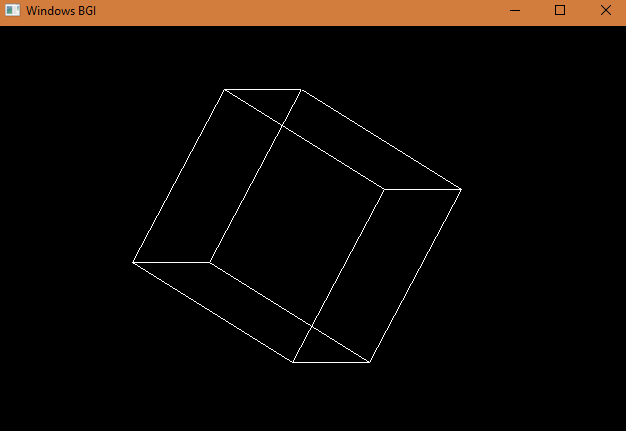
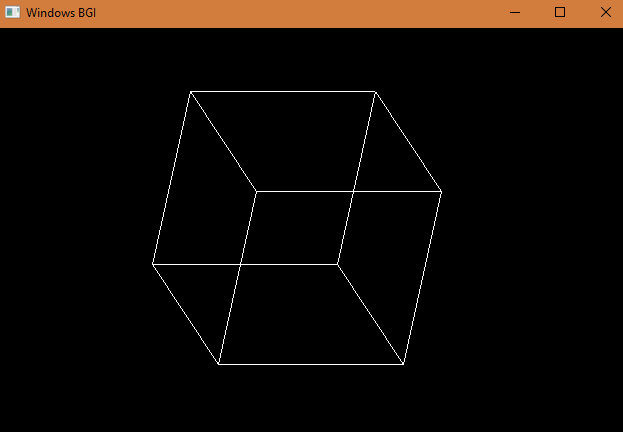
}

# OUTPUT APPLICATION

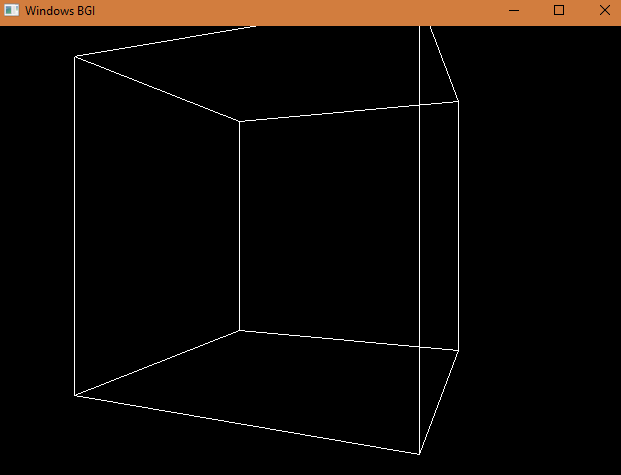
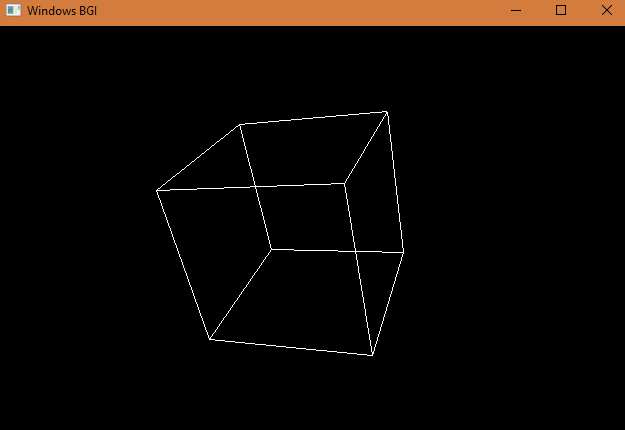
INTERFACE MENU:



ORTHOGRAPHIC PROJECTIONS:



PERSPECTIVE PROJECTIONS:



*Thank you…….*